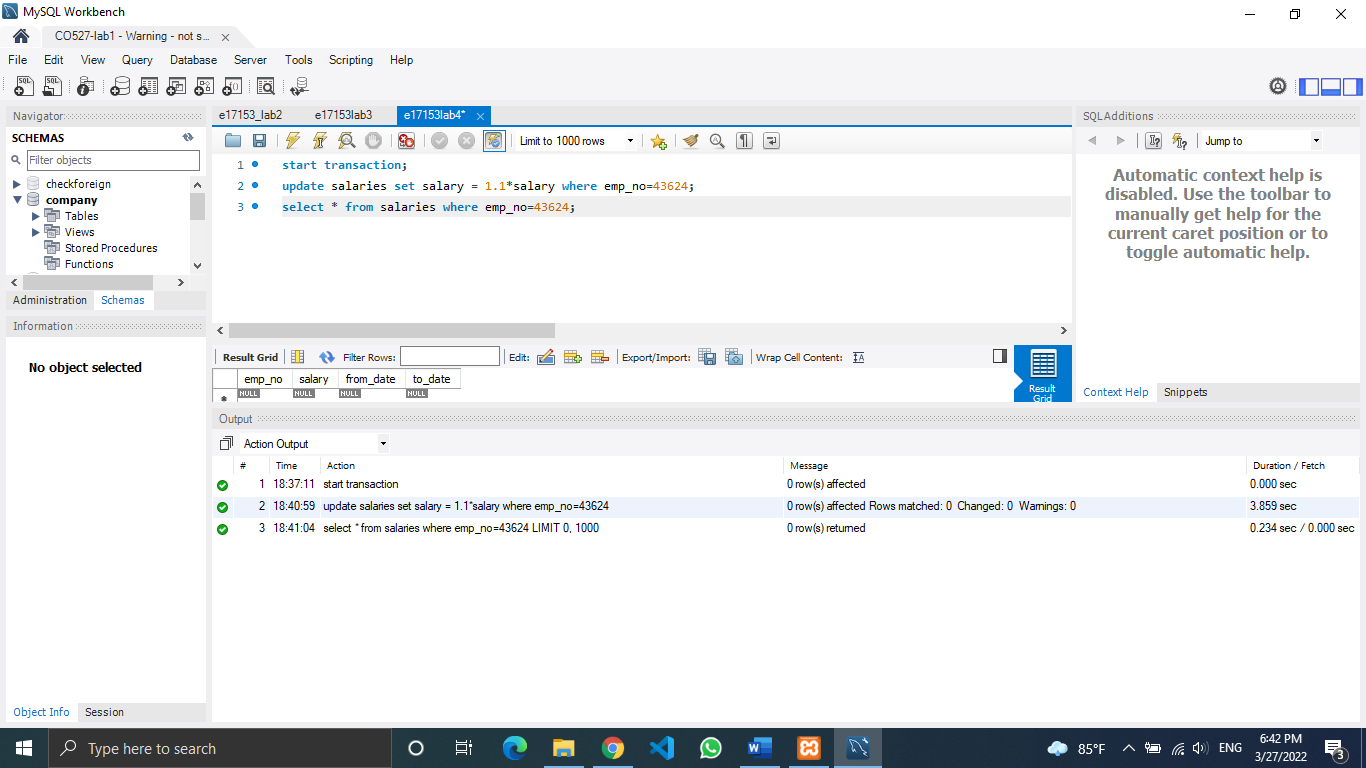
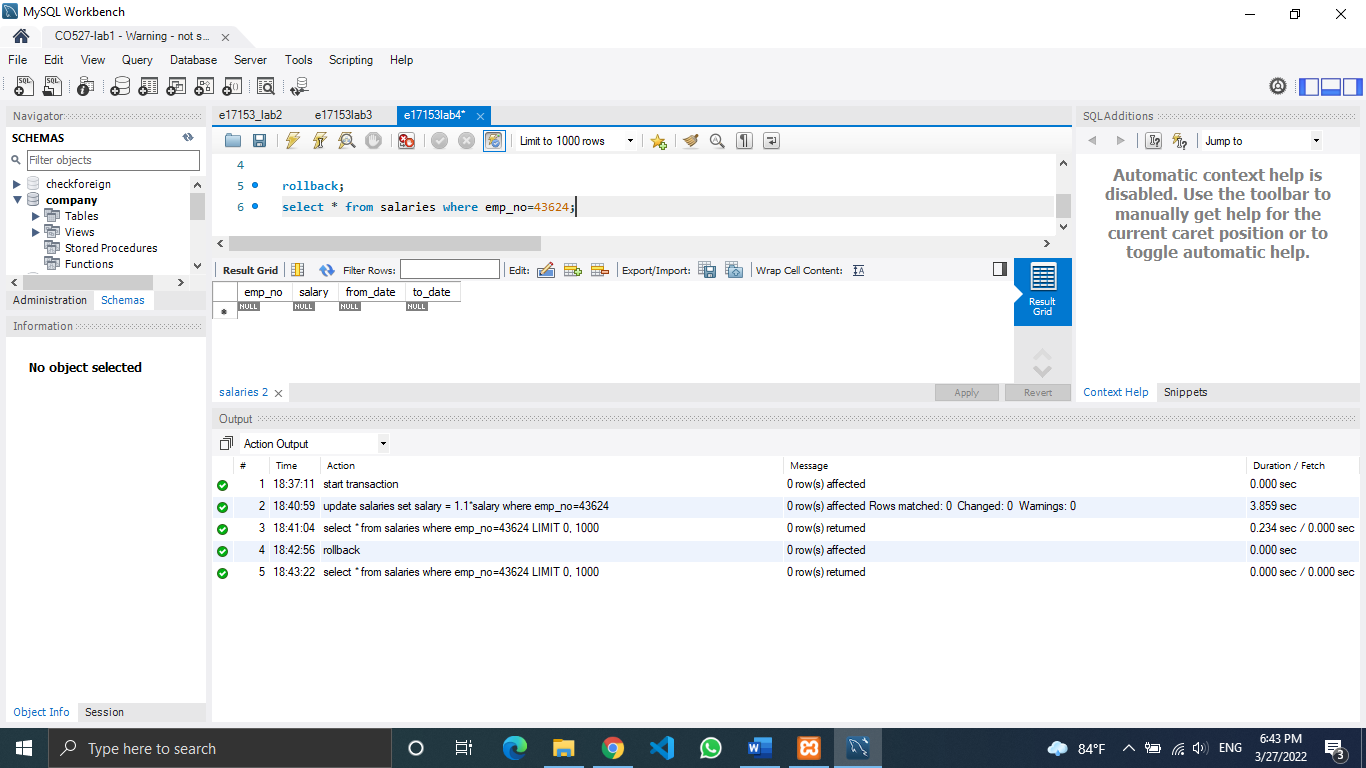
CO527 – Advanced Database Systems

Lab 4 – Transaction Processing

Karunachandra R.H.I.O.

E/17/153

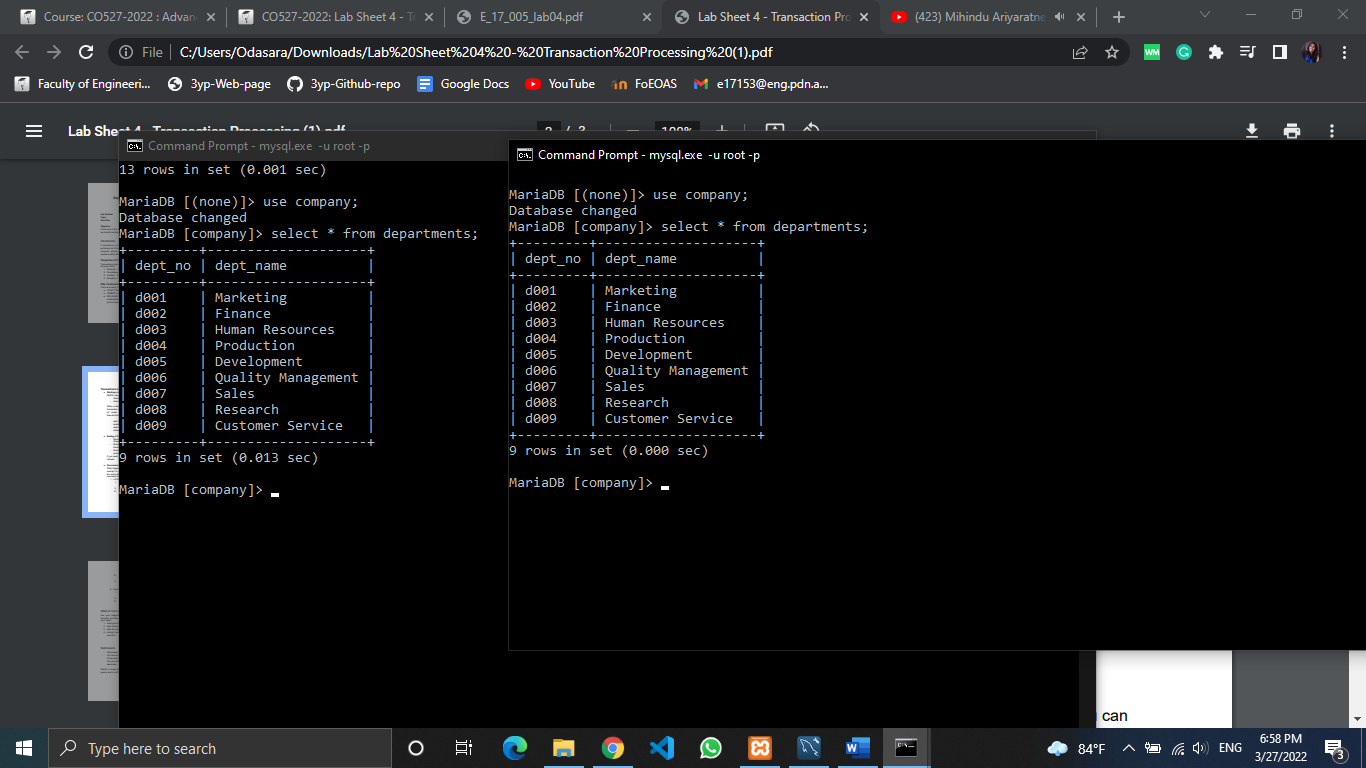




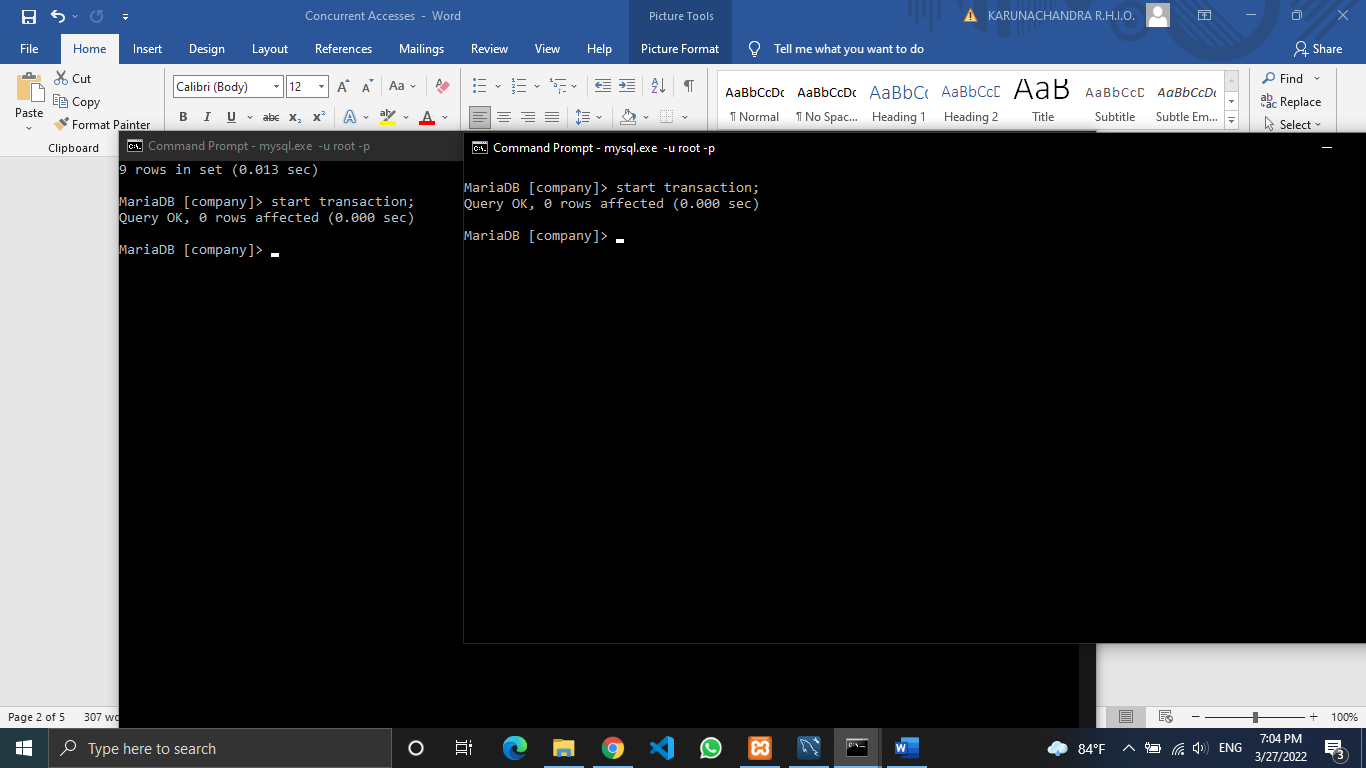
Concurrent Accesses

I of ACID

1. Issue a select query to view the current status of the department table in both sessions.



1. Now, start transaction running start transaction in both sessions.

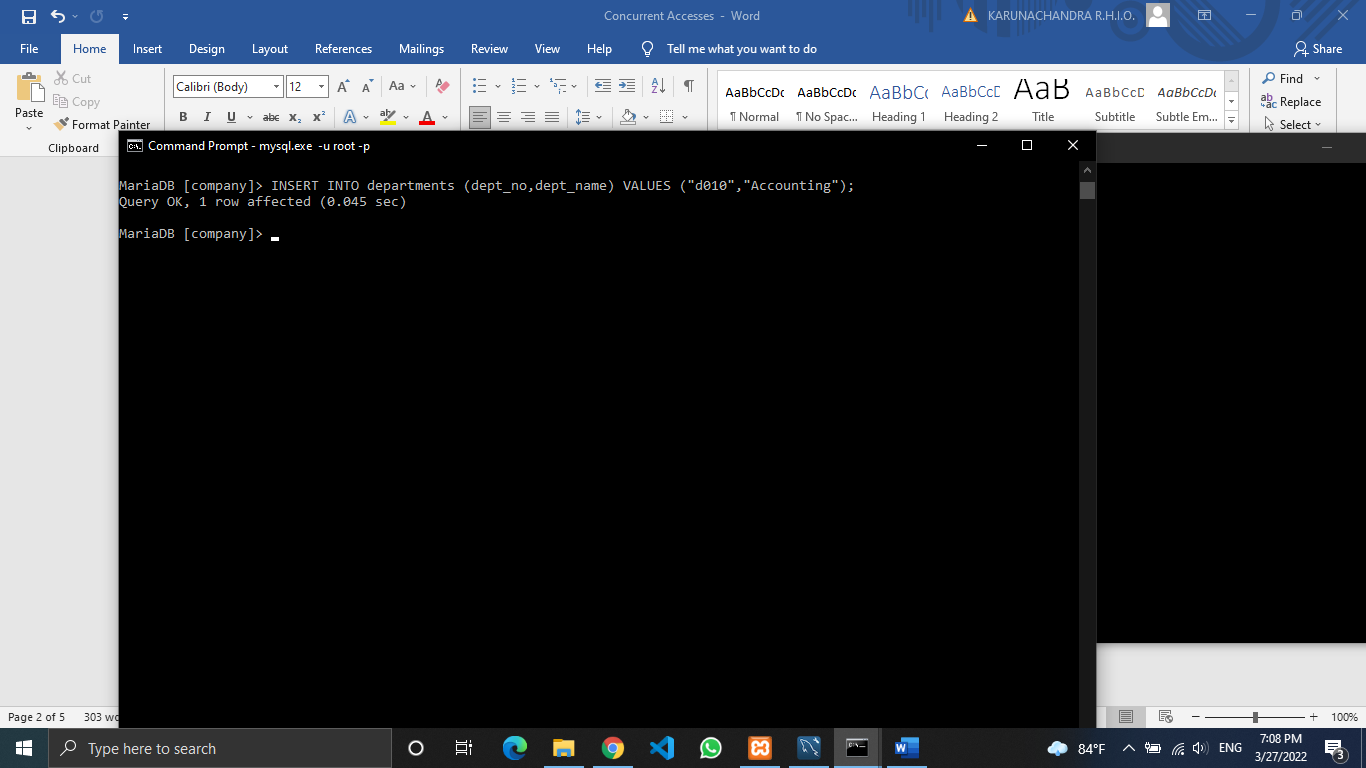


1. Insert a new row into the departments table from the 1st session

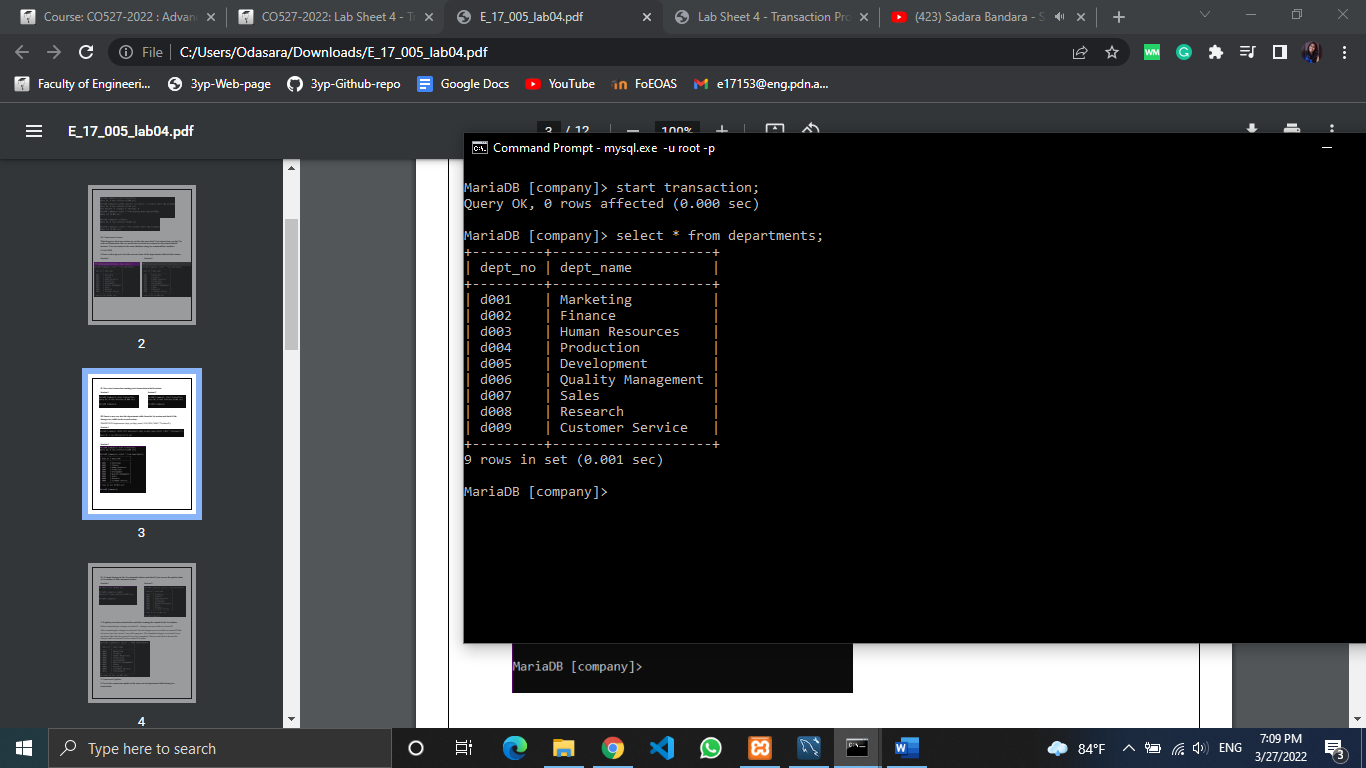
and check if the changes are visible in the second session.

INSERT INTO departments (dept\_no,dept\_name) VALUES ("d010","Accounting");

Session 1



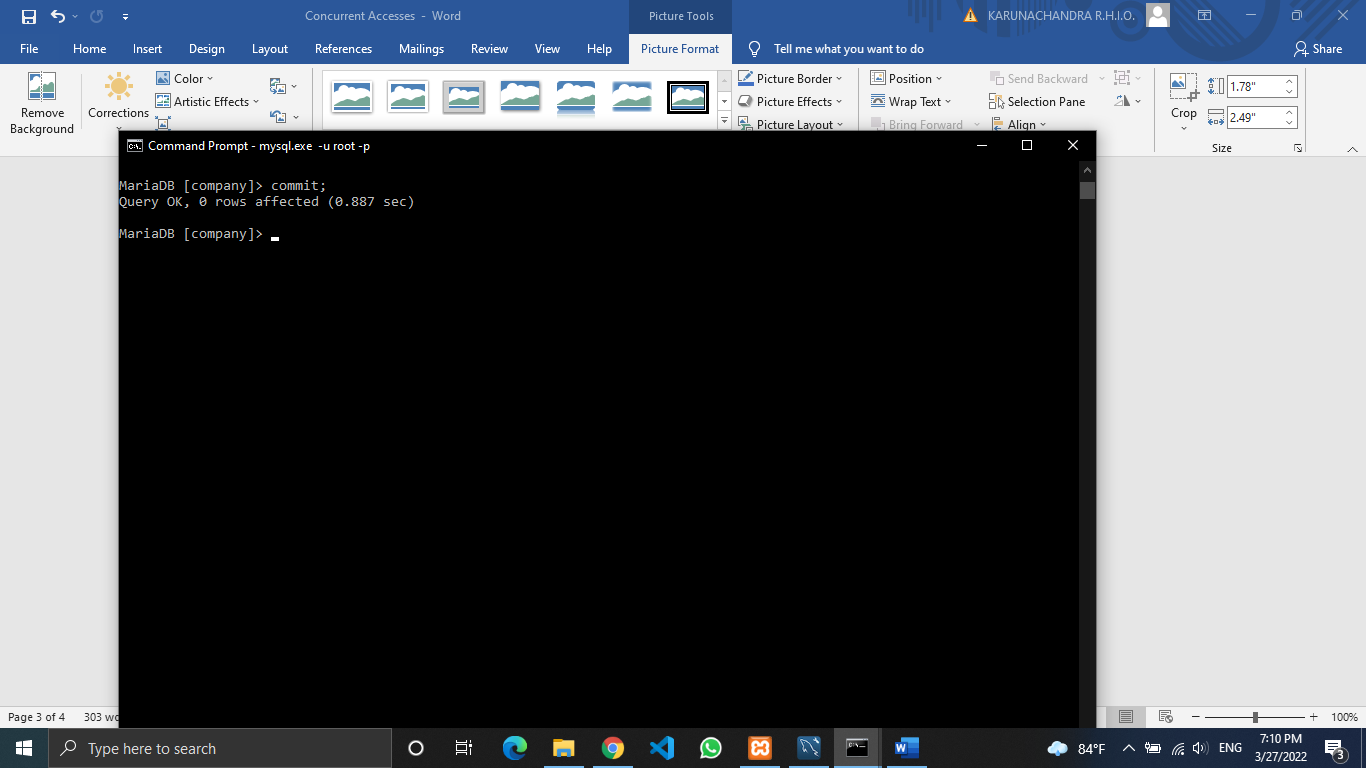
Session 2



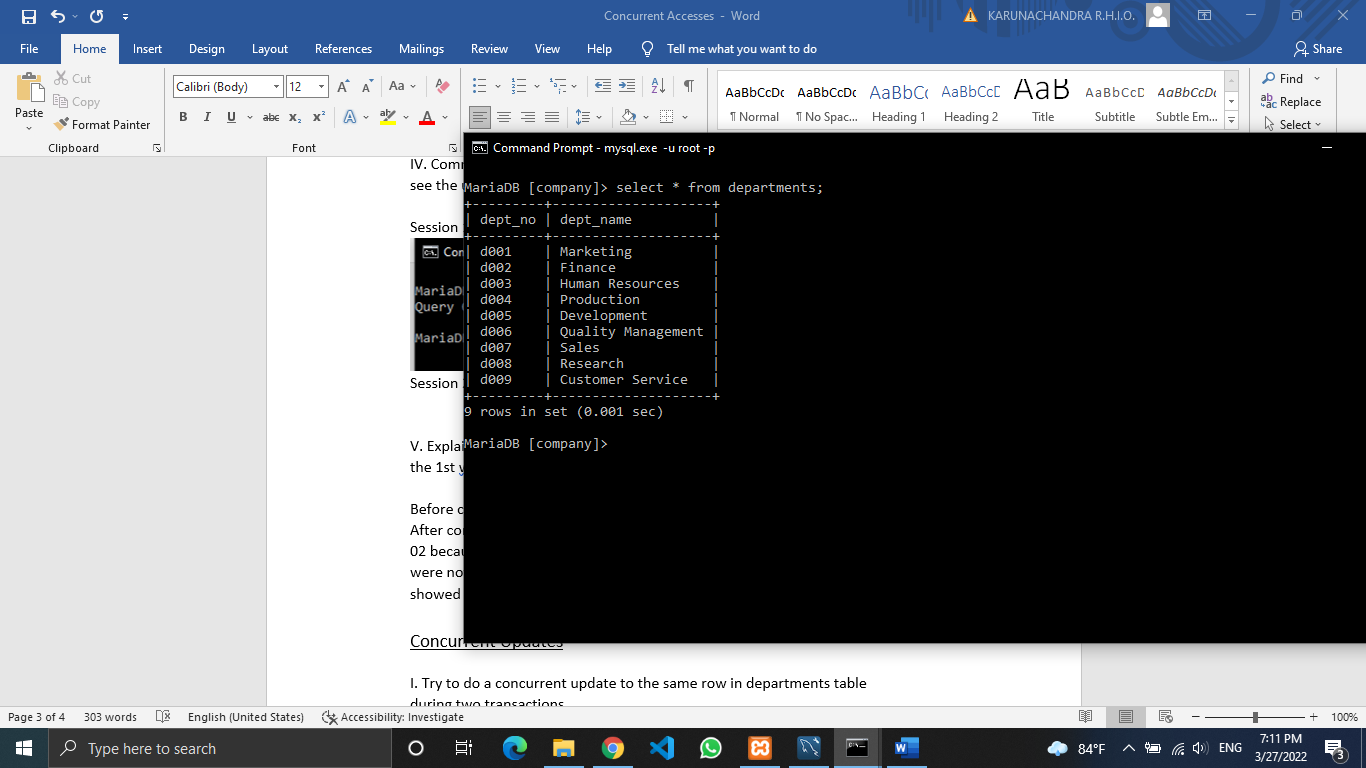
IV. Commit changes in the 1st command window and check if you can

see the updates done at 1st window in 2nd command window.

Session 1



Session 2



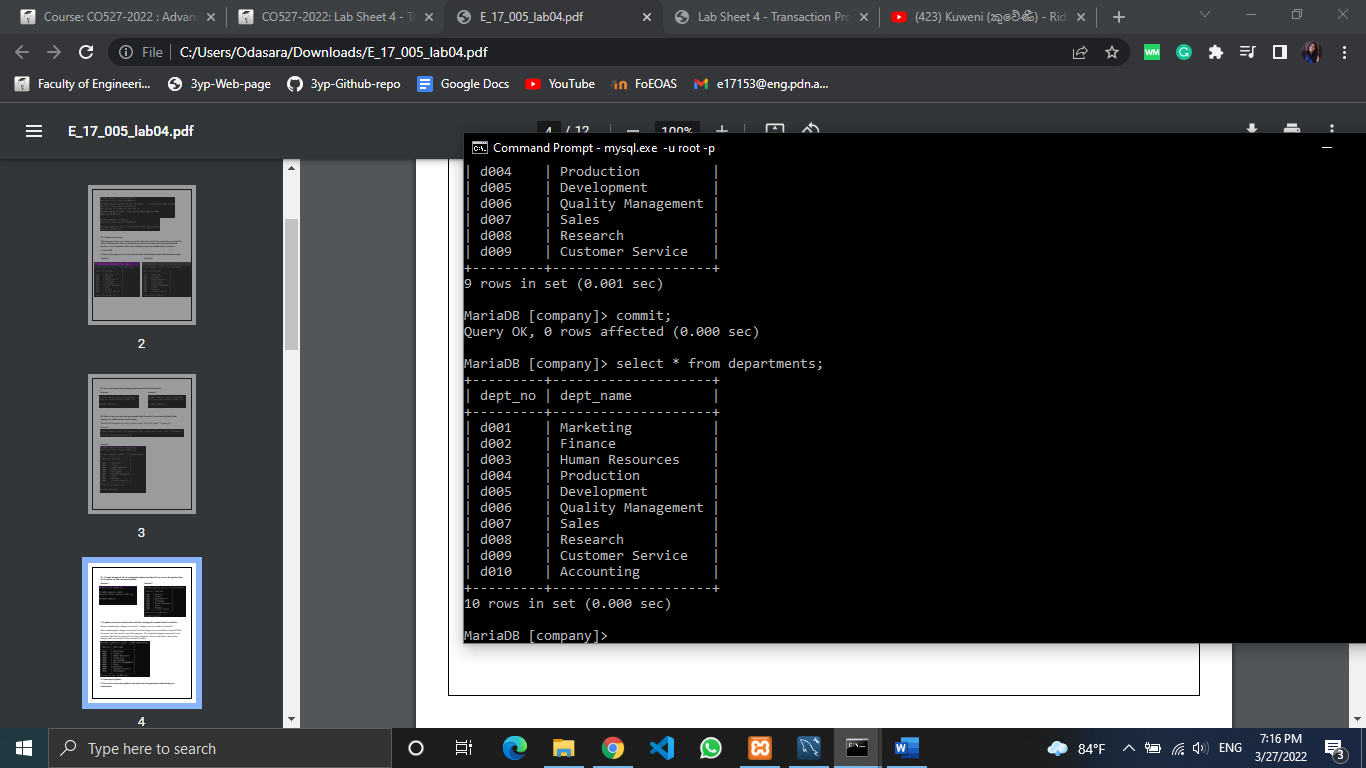
V. Explain your observations before and after running the commit in the 1st window.

There was no change in the session 02 output.

Before committing the changes in session 01, session 02 changes were not visible.

After committing the changes in session 01 also the changes were not visible in session 02.

This was because the session 2 was still in progress. ( Didn’t end the transaction yet.) After the commit of session 02, the changes could be seen.

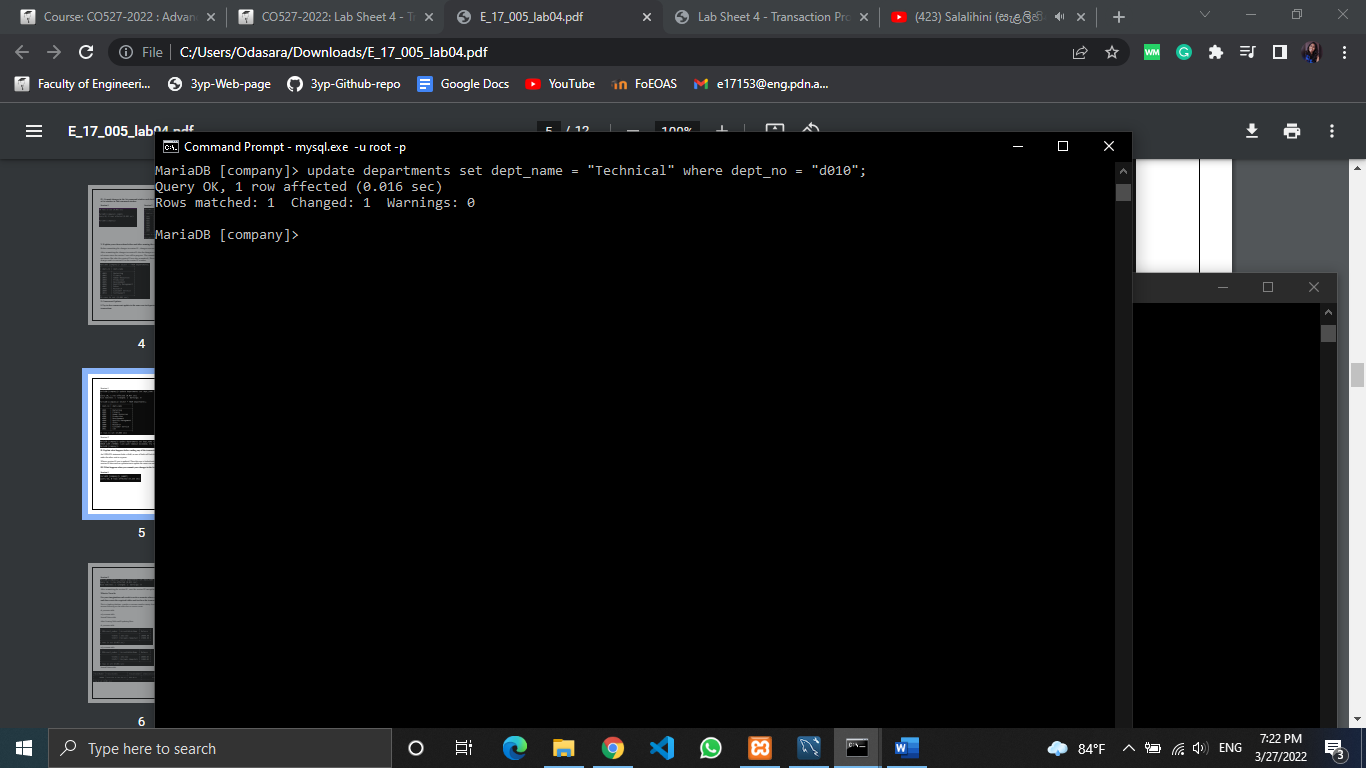


Concurrent Updates

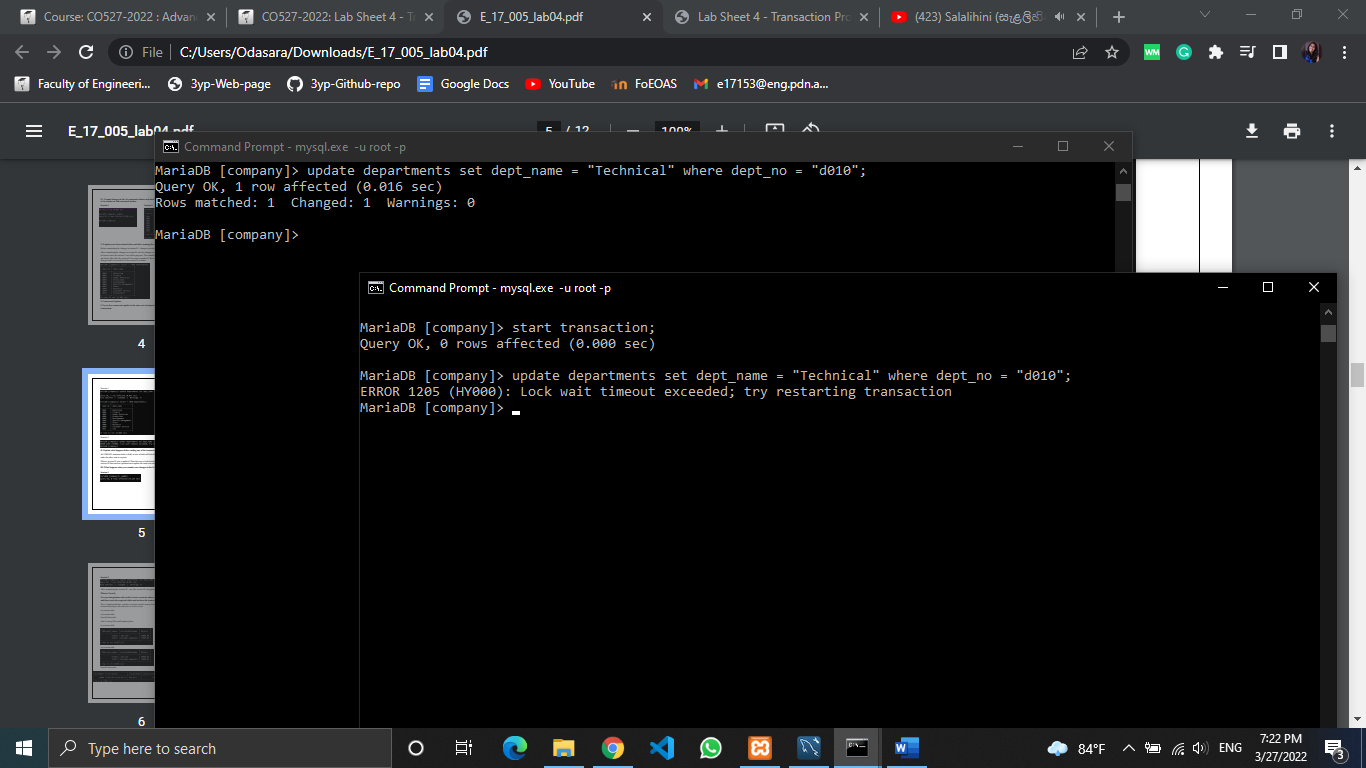
I. Try to do a concurrent update to the same row in departments table

during two transactions

Session 1



Session 2



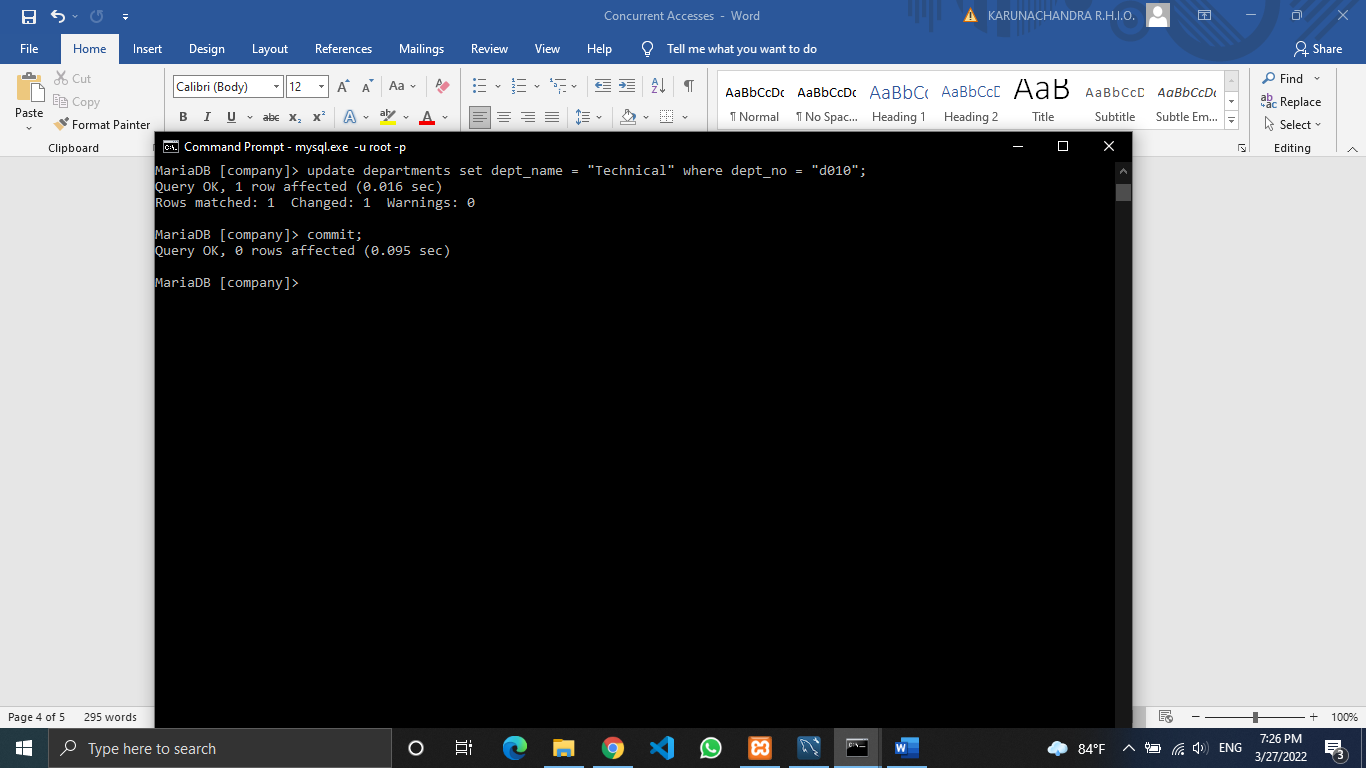
II. Explain what happens before ending any of the transactions.

An UPDATE statement locks a field, so the session 01 locked the field to be updated, and made the session 02 wait in a queue.

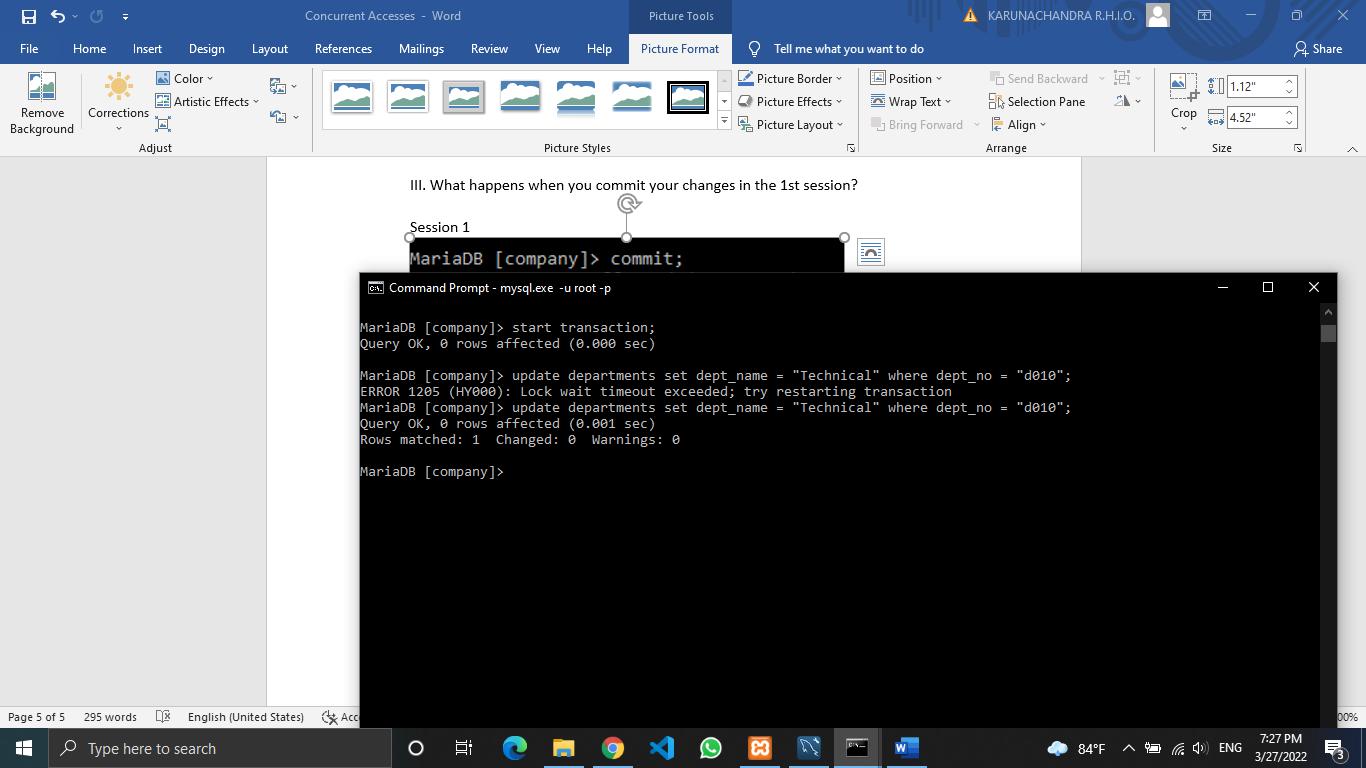
When a session 01 row is updated. Then this row is locked until the commit is done. So, the session 02 does not have permission to update the same row unless session 01 commits.

III. What happens when you commit your changes in the 1st session?

Session 1



Session 2



After committing the session 01, it is possible to update the same row in the session 02.

What to Turn In

Use your imagination and words to write a scenario where using transactions is essential and then create the required tables and test how the transaction will effect your tables,

1. during the transaction execution.

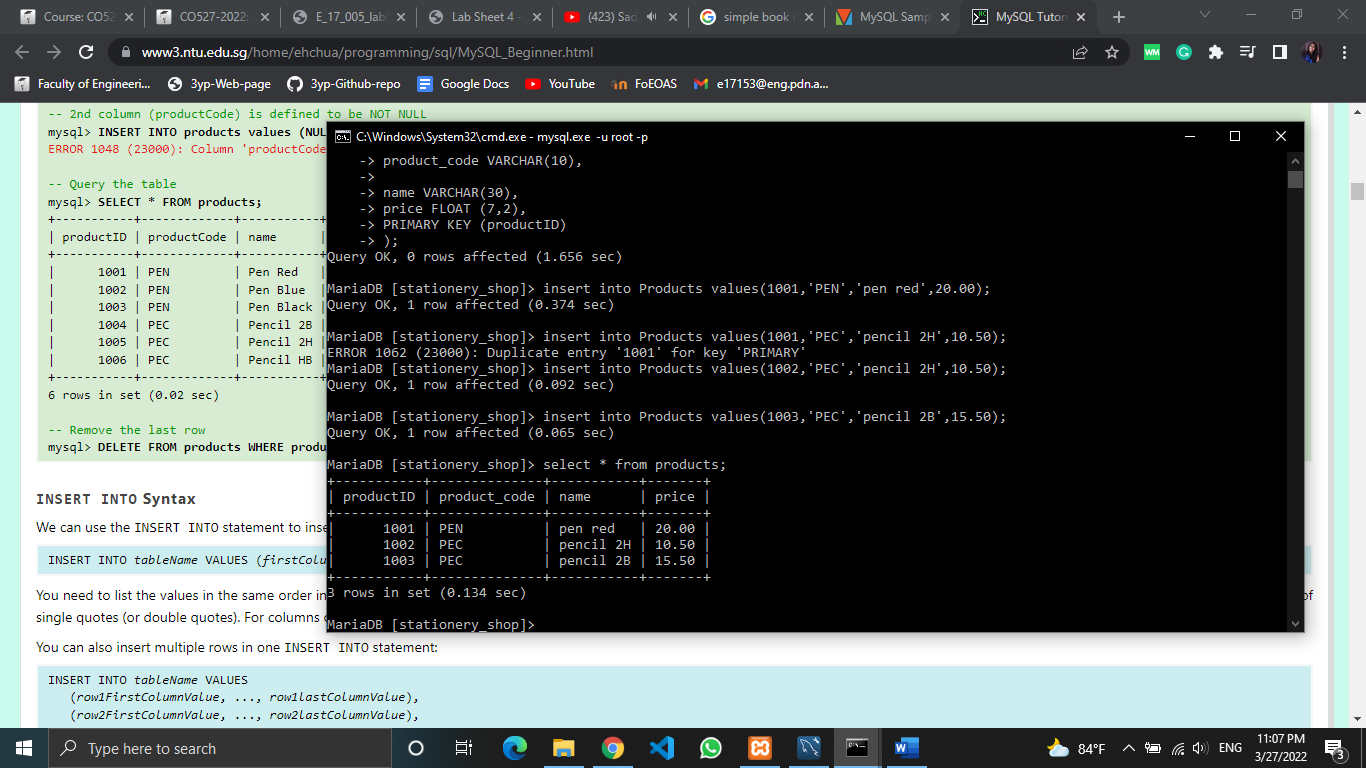
2. after rollback statement.

3. after the commit statement.

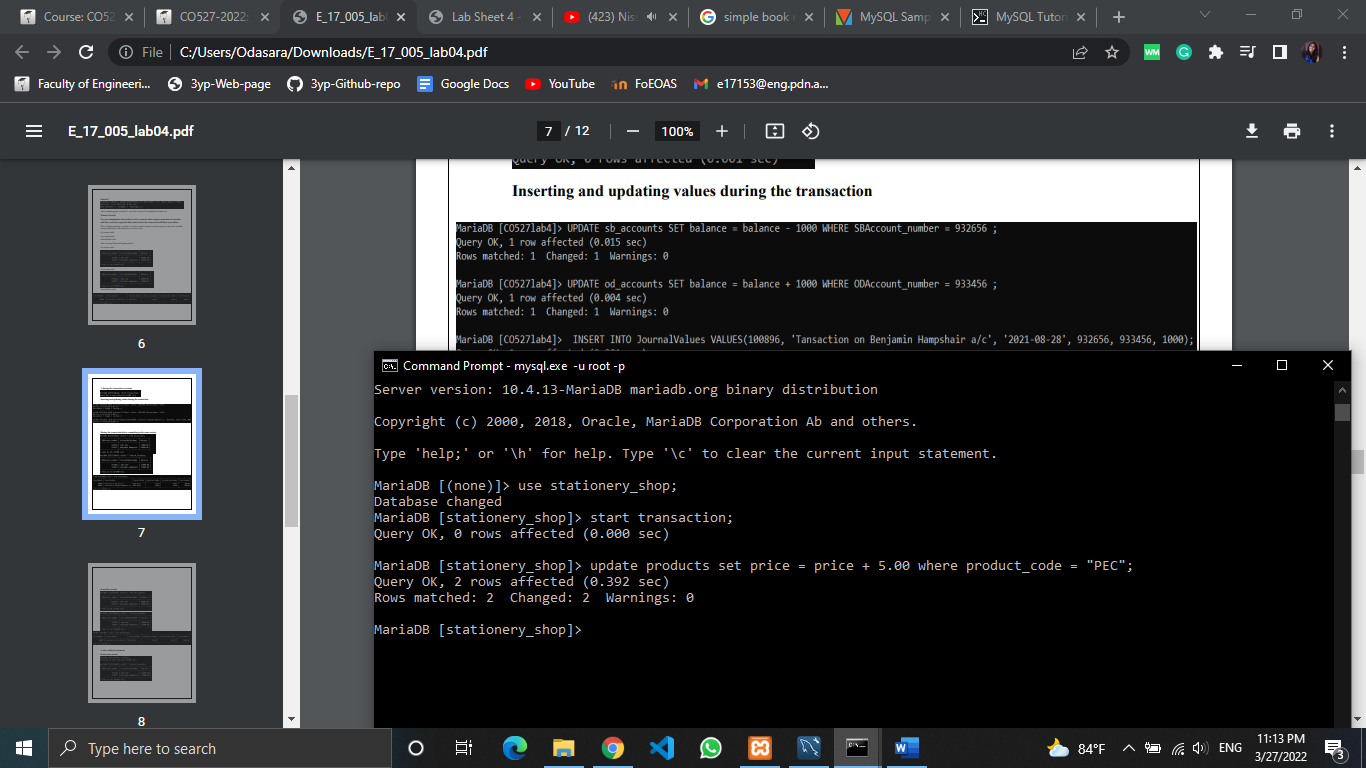
4. during 2 concurrent transactions, both of them update a record and both of them commit it.

Let’s create a simple database for a stationary company.

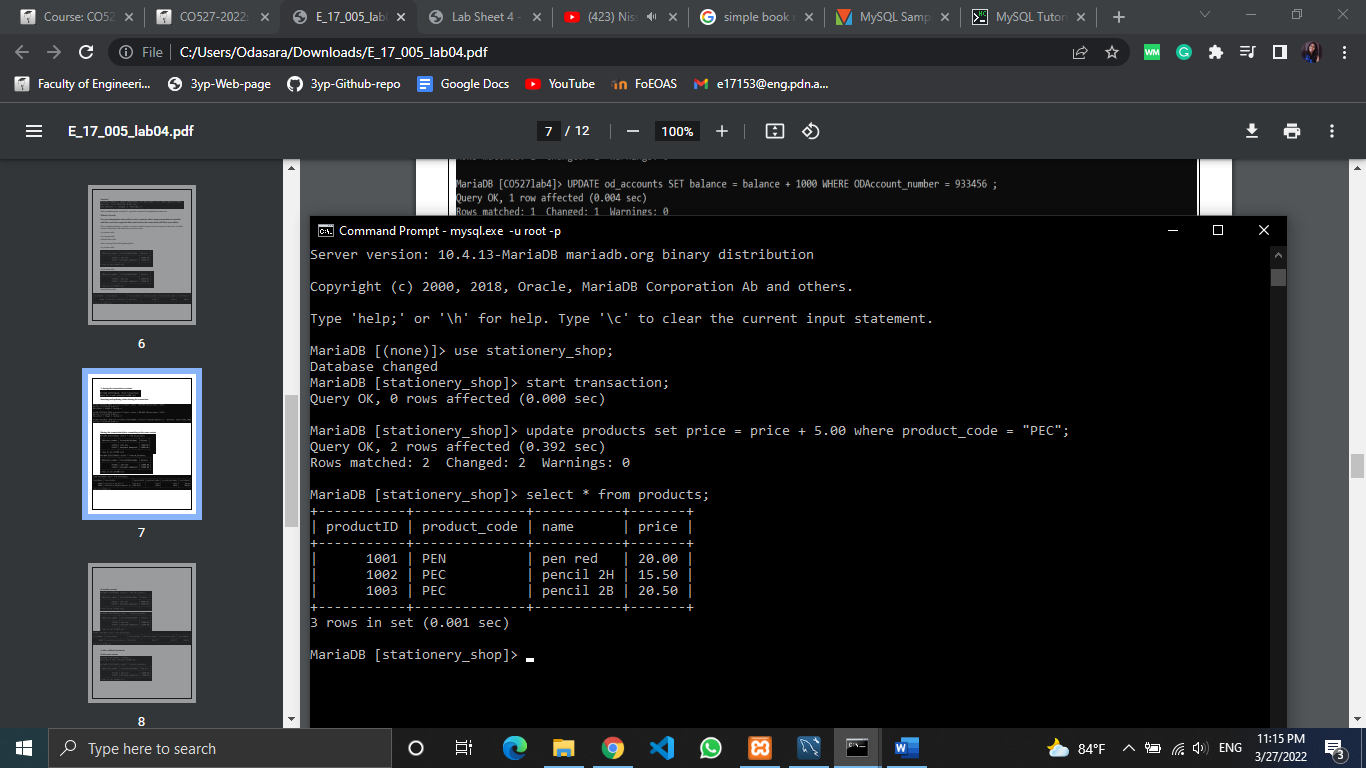
Here is the ‘product’ table of the dataset, ‘stationery shop’.



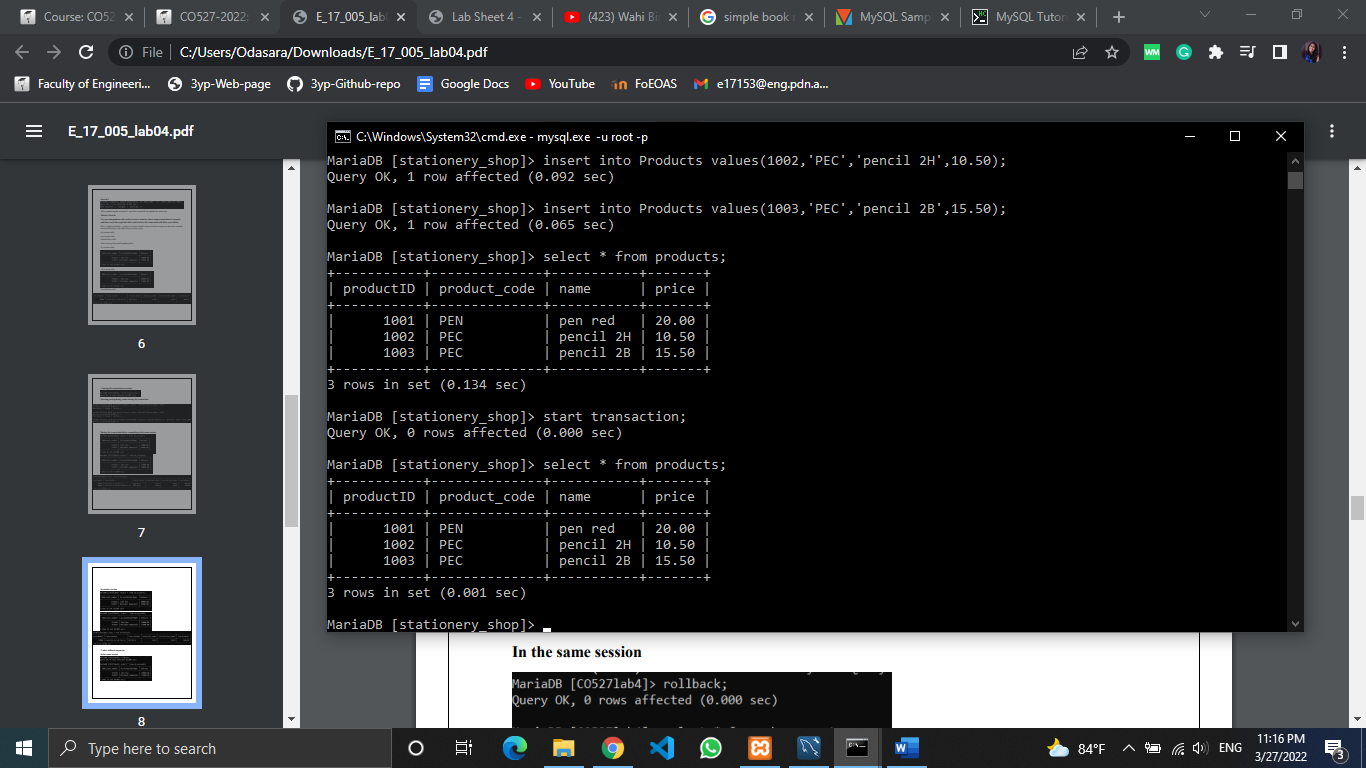
1.Transaction

In the session 01, an update was done.

Session 01 before commiting (Updates can be seen)

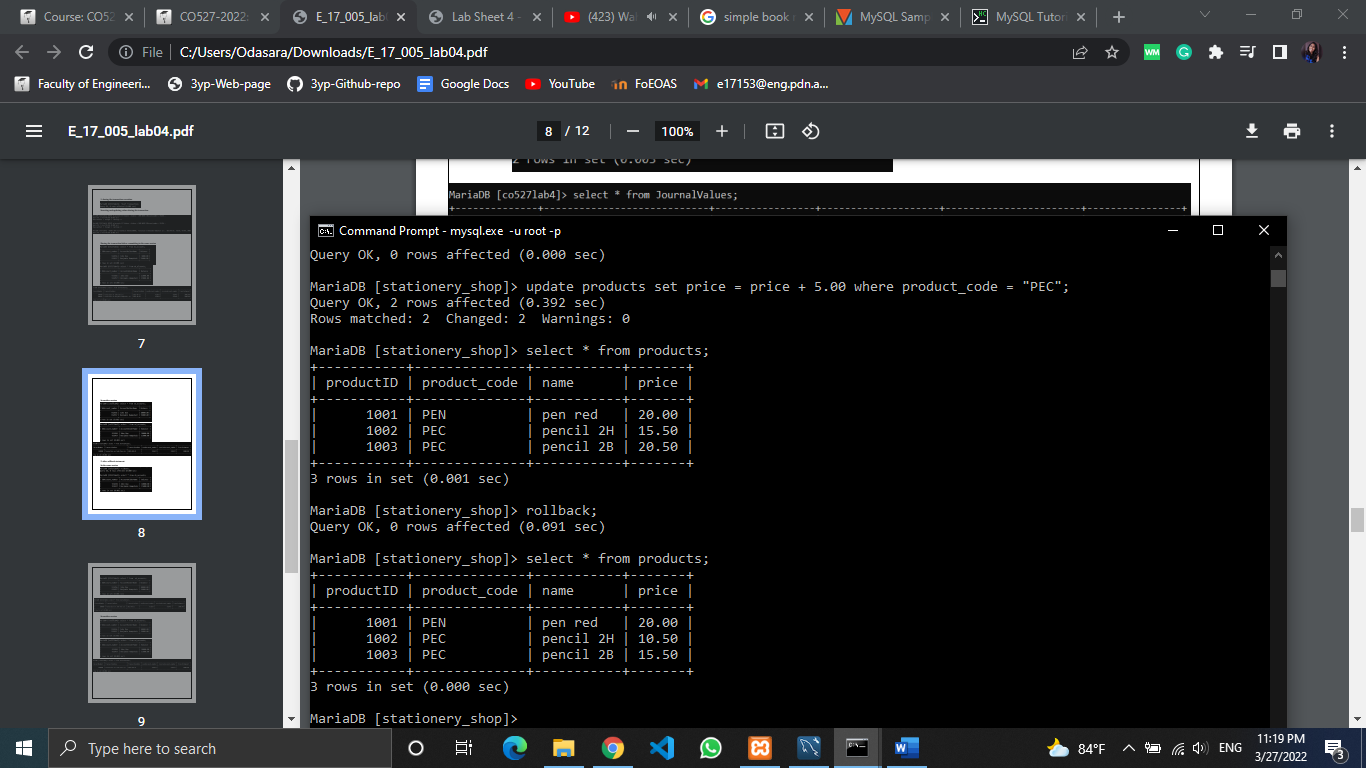


Session 02 (Updates are not yet applied)

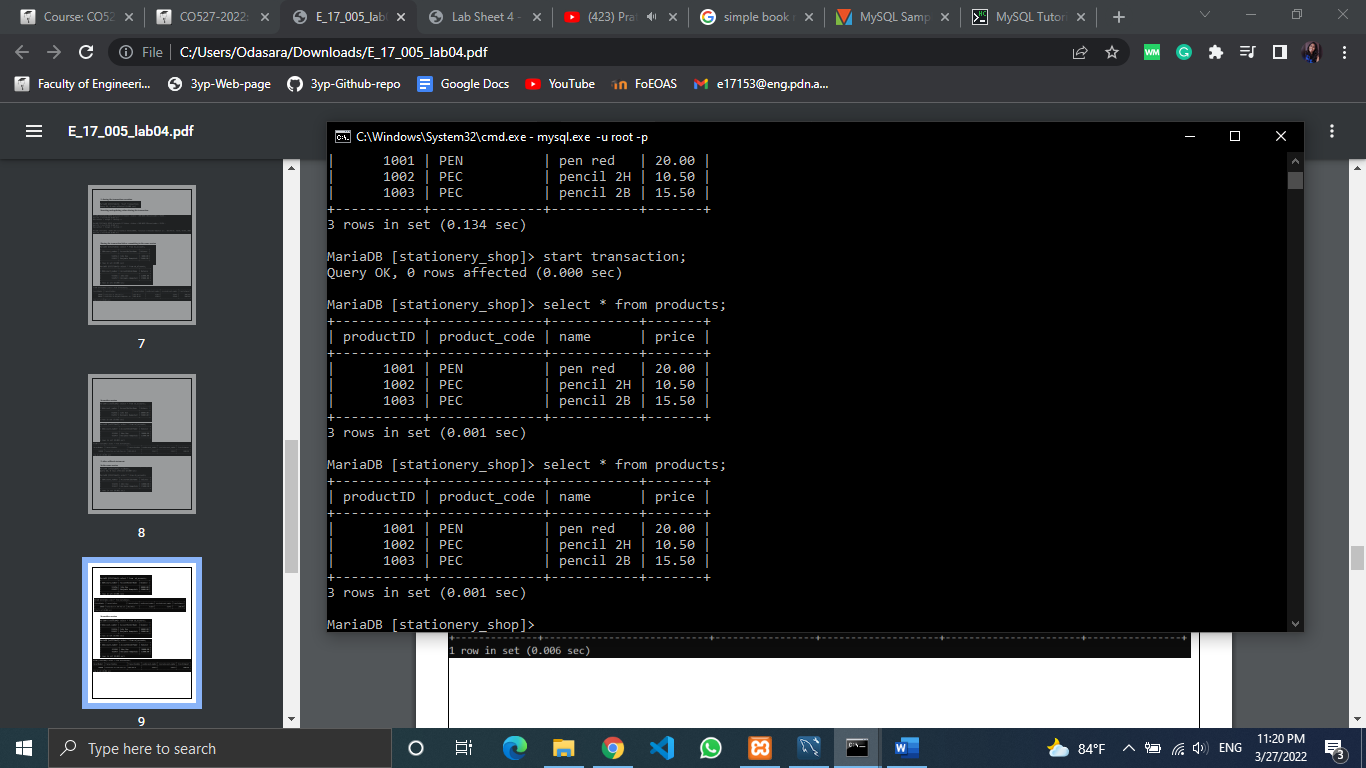


2.Rollback

Session 01 (Updates have been reversed)

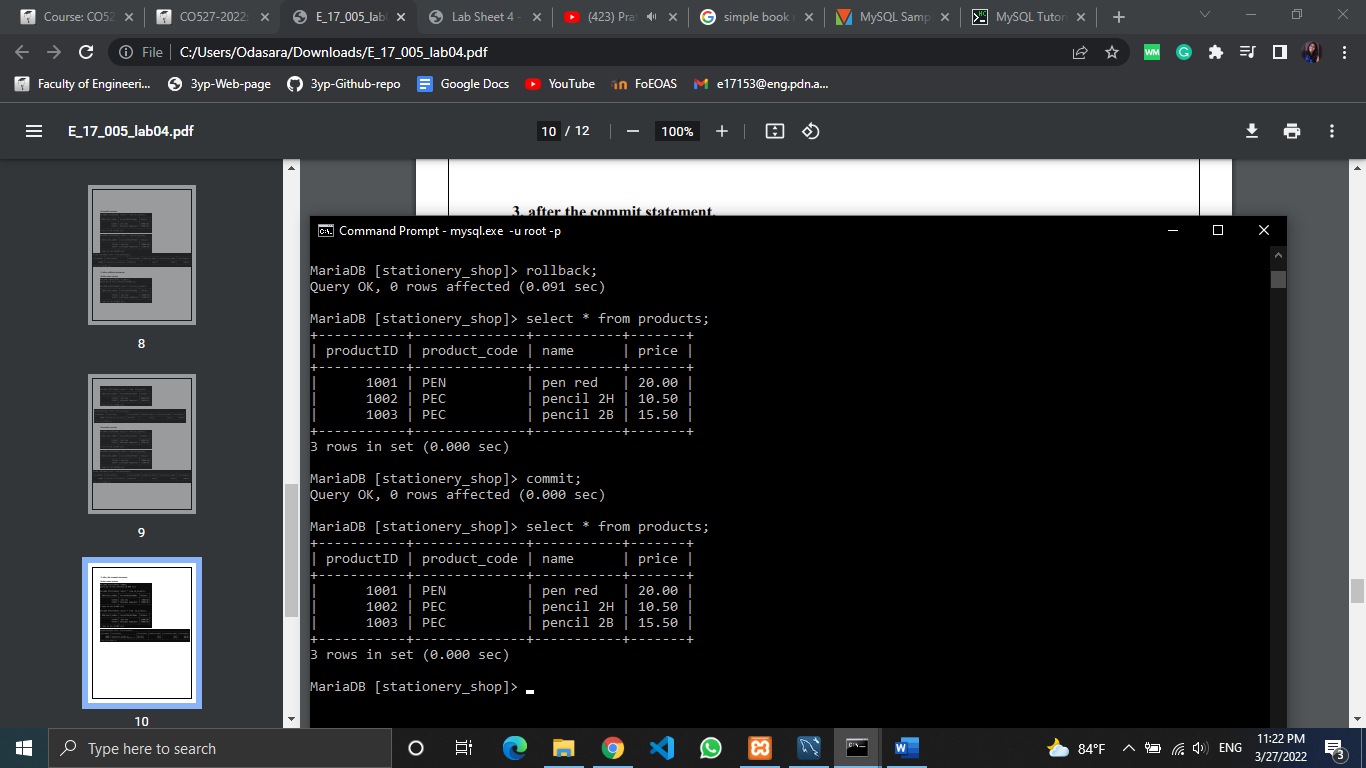


Session 02 (No changes)

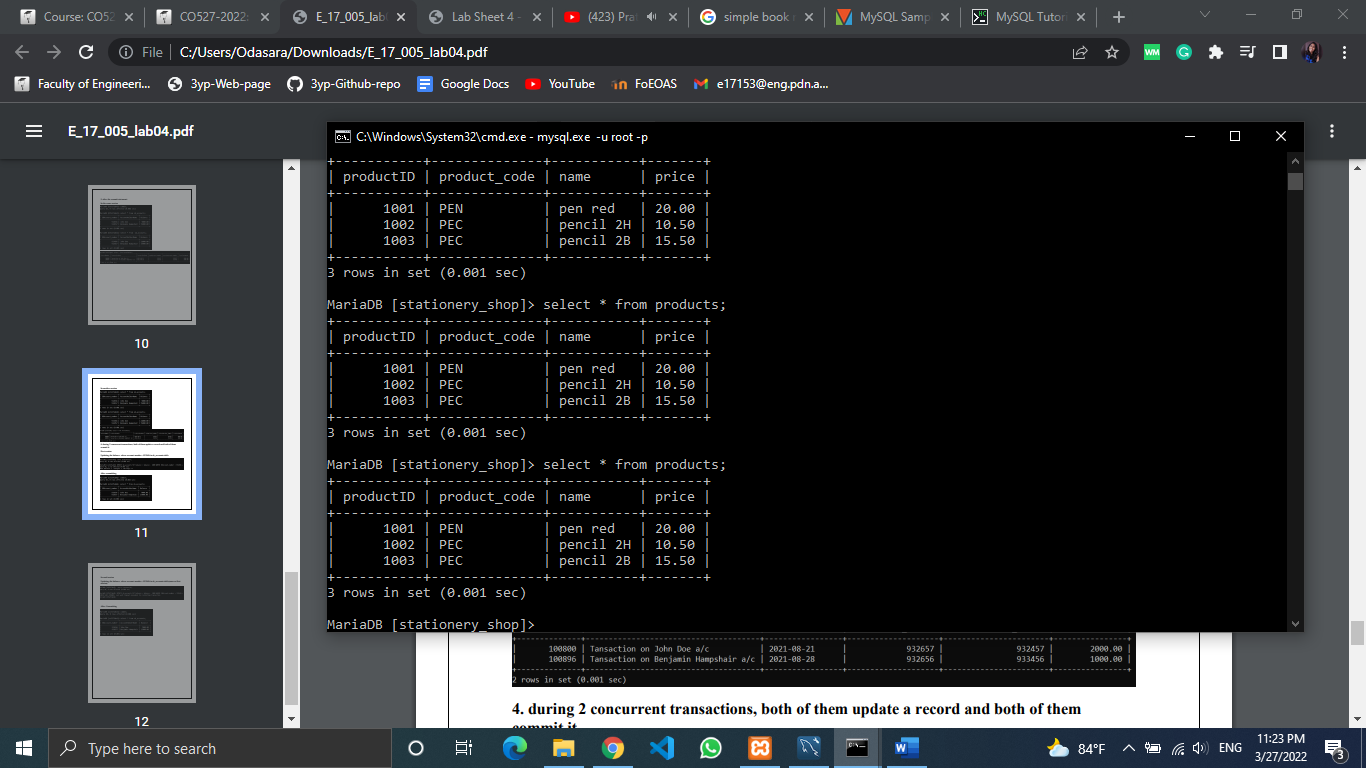


3. Commit

Session 01 (In the previous status- no updates)

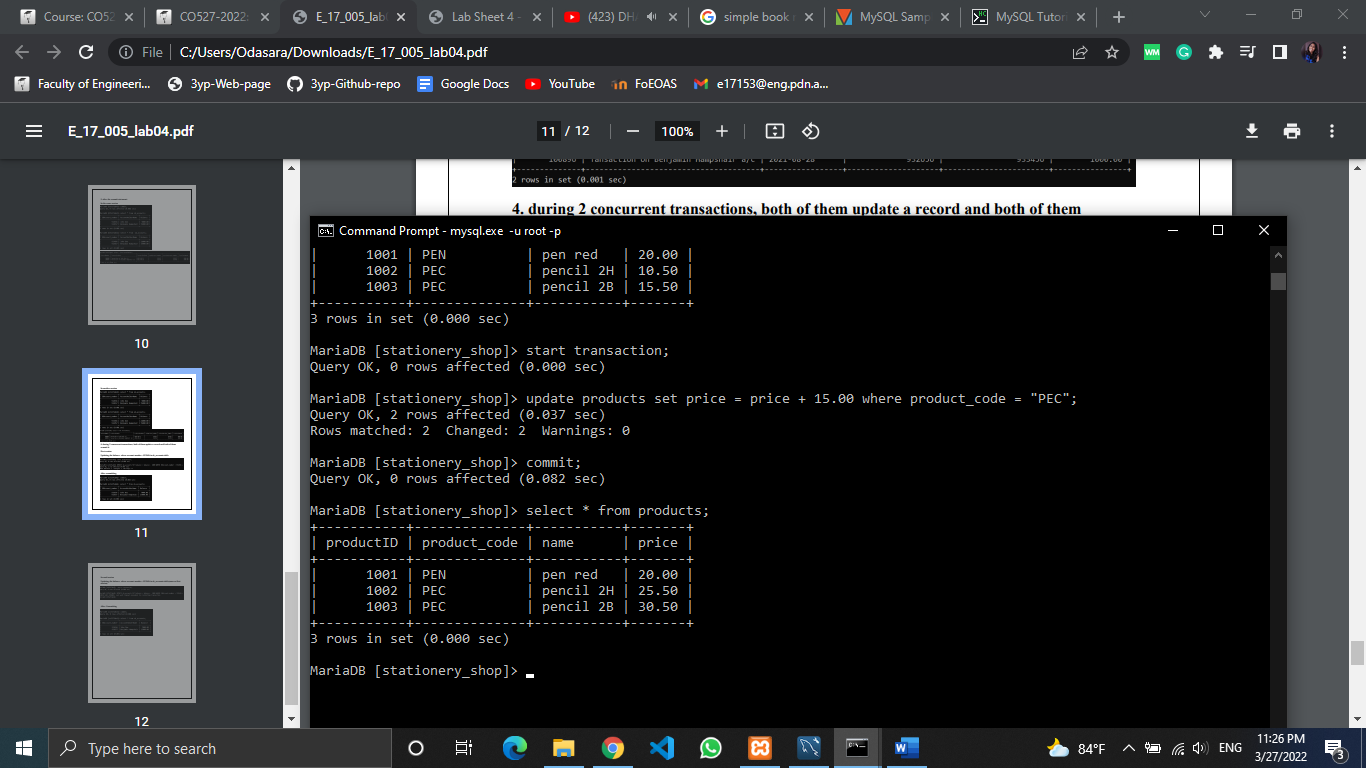


Session 02 (No changes)

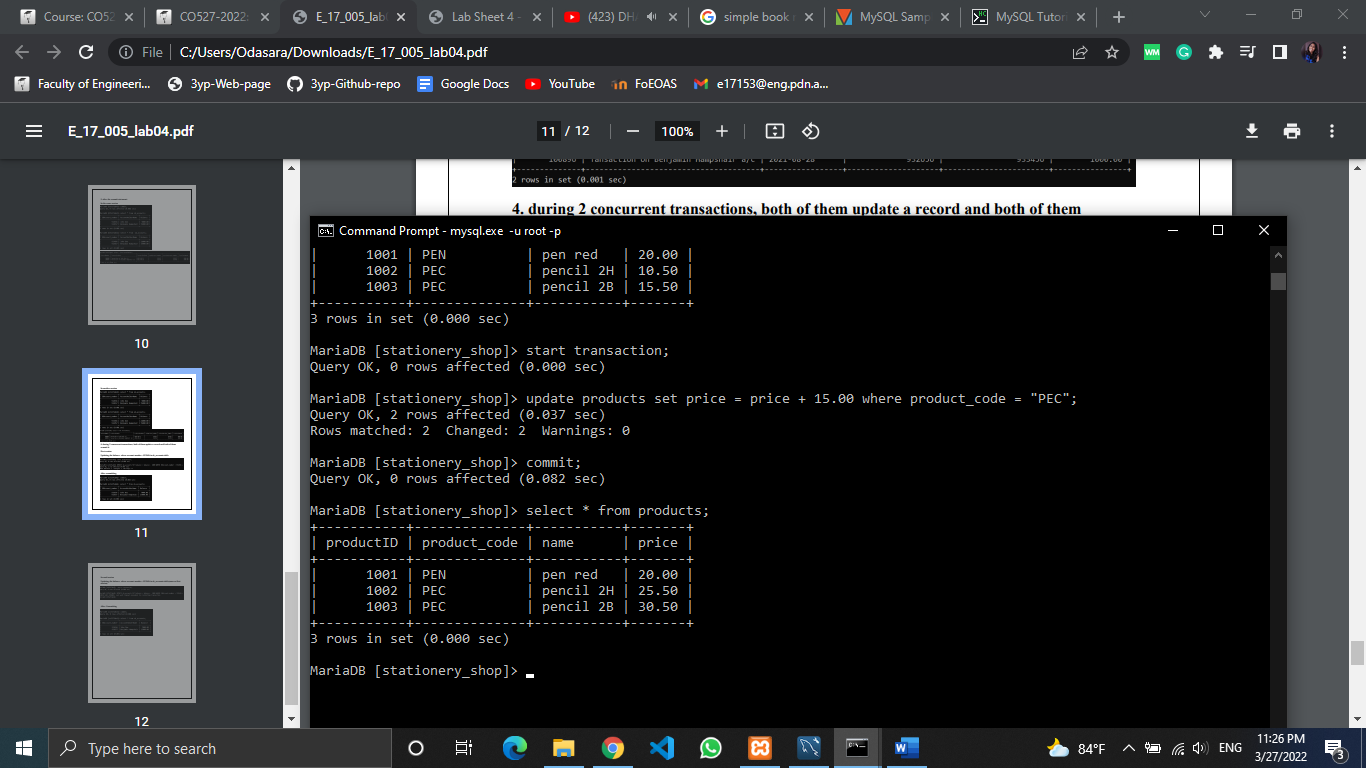


4. during 2 concurrent transactions, both of them update a record and both of them commit it.

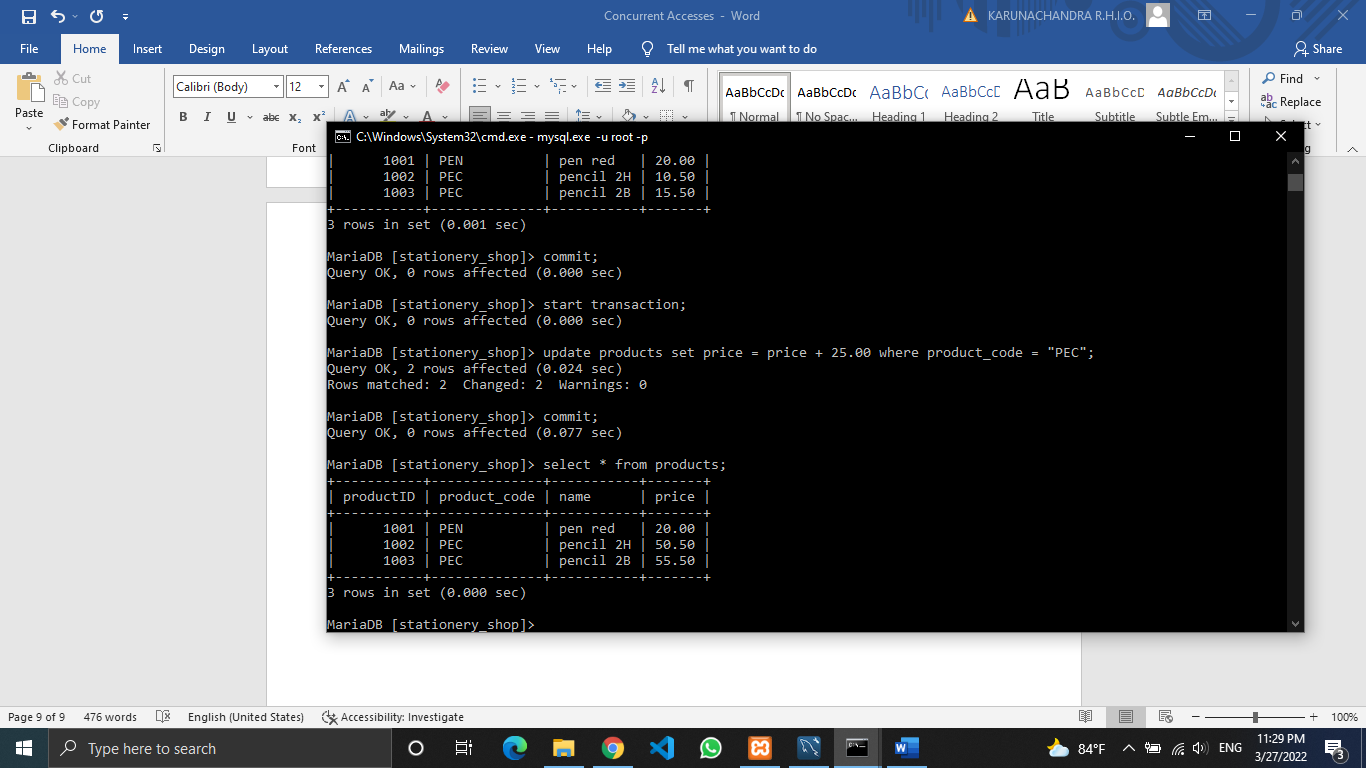
Session 01 – update the prices of “PEC” items



Session 01- commit



Session 02- update the same row



Session 02- commit (New values were updated)

